IN THE DRAWINGS

Figure 6 has been amended as indicated in red in the attached annotated marked-up drawing. A replacement sheet of Figure 6 is also attached.

REMARKS

The objection to the drawings, set forth in paragraph 5, has been corrected.

The objections set forth in paragraph 7 have been corrected as suggested.

Reconsideration of the rejections of the claims, based on Shaylor, is respectfully requested. Shaylor does not seem to have anything to do with querying metadata. Nothing seems to suggest this in any of the cited material. (Incidentally, it is believed that the citation in the text to column 3, lines 9-23, is actually intended to refer to column 5, lines 9-23). Moreover, there is no limiting a search scope within a local memory sub-region.

The material in column 5 says that a received byte code is invoked and this may generate a lookup in a constant pool 206. It appears that the constant pool in Figure 2 is a lookup table.

As explained in the background, when querying metadata in the past, the typical approach was to use a lookup table which is very burdensome. As explained in column 5 in the cited material, "performing these multiple lookups can be very time consuming." See Shaylor at column 5, lines 19-20.

Thus, it does not appear that Shaylor is querying method metadata and, even if he is, he is not limiting the search scope within a local memory sub-region of the code address, but, instead, is simply using the conventional approach of a lookup table.

Therefore, reconsideration is respectfully requested.

Respectfully submitted,

Date: September 6, 2007

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation

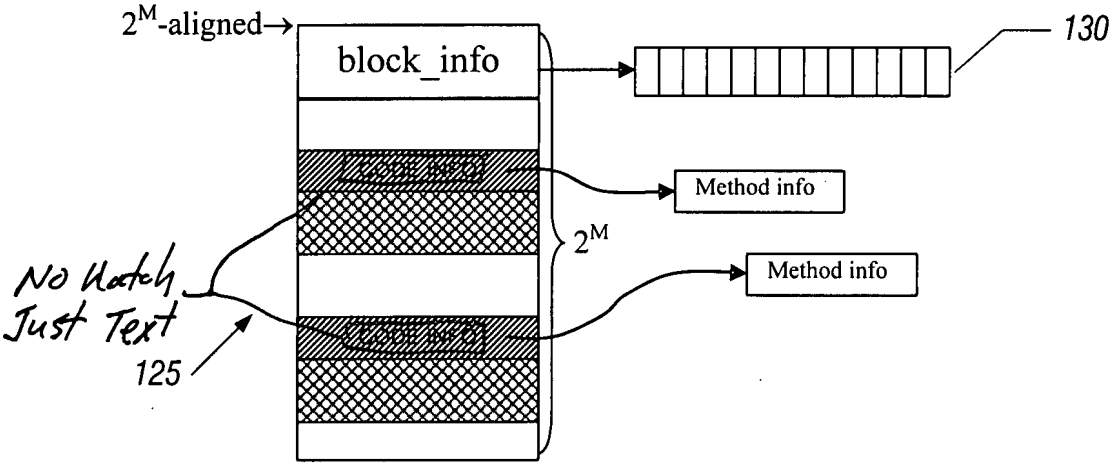

FIG. 6

```
method_info *query_method_info_by_IP(void* ip)
{
   int bit_index = map_address_to_bit_index (A);
   if (bit_index is invalid)
      return NULL;
   bit_index = alloc_bits.last_set_bit_from(bit_index);
   void* code_addr = map_bit_index_to_address (bit_index);
   code_info *ci = (code_info*)code_addr;
   method_info* mi = ci->mi;
}
```

FIG. 7